File Sharing in P2P : Missing Block Paradigmand upload strategies

Fabien Mathieu, Julien Reynier

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

File Sharing in P2P : Missing Block Paradigm and upload strategies

Fabien Mathieu and Julien Reynier

No 5193

Mai 2004

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--5193--FR+ENG

File Sharing in P2P : Missing Block Paradigm and upload strategies

Fabien Mathieu and Julien Reynier

Thème 1 — Réseaux et systèmes
Projet Gyroweb

Rapport de recherche n 5193 — Mai 2004 — 15 pages

Abstract: Unfinished download is a frequent issue in most peer-to-peer download architectures. For some networks, it seems to be an inherent problem, however even "smart" networks can encounter the *whole file but one block downloaded* case, which we call the *missing block paradigm*.

We propose a simple and versatile model of file-sharing that applies to all block-oriented file-sharing protocols used in softwares such as MlDonkey or BitTorrent. Simulations using this model show that *missing block* can occur even with a popular file, and lead to some theoretical explanation.

These fundamental results offer a new understanding of downloading issue in file-sharing networks, and show new strategies existing protocols could use.

Key-words: P2P, file-sharing, download issues, upload strategies

(Résumé : tsvp)

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Persistance et diffusion des fichiers dans les réseaux pair-à-pair

Résumé : L'impossibilité de finir de télécharger des fichiers est une situation fréquente dans les réseaux pair-à-pair de partage de fichiers. Si parfois ce problème semble intrinsèquement lié à la nature du réseau, il semble que même des réseaux *intelligents* vis-à-vis du téléchargement peuvent aboutir à une situation où toutes les parties du fichier sont disponibles sauf une.

Nous proposons un modèle simple et évolutif de partage de fichiers qui peut s'appliquer à tous les protocoles de téléchargement par blocs, comme ou BitTorrent. Les simulations montrent que le cas du *fichier manquant* peut se produire même pour des fichiers populaires, et donnent quelques pistes théoriques.

Ces nouveaux résultats permettent une autre approche des problèmes de téléchargements dans les réseaux de partage de fichiers, et de nouvelles stratégies sont proposées.

Mots-clé : Pair-à-pair, partage de fichiers, problèmes et gestion de téléchargement

1 Introduction

A P2P file-sharing network is an interface that permits the exchange of data between users arriving and departing independently. P2P issues can be classified in three categories:

- dynamical management of subjacent overlay networks[8, 9, 6, 4];
- publication and search of shared content[5, 2];
- downloading protocols.

The last domain, downloading protocols, seems less studied than the two others. However, countless people have downloading issues every day. In this paper we focus on the "unfinished download" case.

For example, imagine you want to download your favorite linux distribution. For more efficiency, you use three of the most popular file-sharing softwares, KaZaA, MLDonkey MlDonkey[3] and BitTorrent[1]. Downloads start, no problem to be seen, you leave for a week. When you come back, none of your downloads is finished, downloading is null, all you got are those messages:

KaZaA `415312kb/714204kb downloaded;`
`more sources needed`
eDonkey `64/65 parts downloaded;`
`last seen complete: a long time ago`
Bittorrent `99,7% downloaded; connected to 0 seeds;`
`also seeing 0.997 distributed copies`

The KaZaA message is not so surprising knowing that KaZaA peers only upload finished downloads. As long as there exists a user sharing the whole file, downloads go on [1], but once this (these) user(s) quit(s), all is over. The cases of eDonkey and Bittorrent are more interesting. To allow partially downloaded content to be shared, these protocols broke files into smaller blocks. Experience shows it is not just bad luck if the download stagnates at the last block.

In the next section, we introduce the approach and the different assumptions used in our model. Section 3 shows interesting results coming from simulations of the model. In section 4 we present some stability results for simulations of 3. This highlights a frequent issue in real file-sharing networks, called *missing block* issue. Section 5 compares the different upload strategies and gives characteristics of safe states, where the data can potentially survive forever...

2 Model

The whole point of this article is to study the sharing of a single file in a totally connected overlay network, which we call *torrent*. We suppose the implicite existence of a server that

1. Note that as we will see, downloads in KaZaA are less efficient than in the two others which allow partially downloaded content to be shared.

organizes the users but does not possess the file itself. This fits well Bittorrent protocol, as users must connect to a so-called *tracker* to participate. *Peers* are users that want to download the file, and that can potentially share partial content. *Seeds* are users that share the whole file. To resume, a torrent is made up of S seeds and P peers trying to get a file split in K blocks.

Study of such strategies is often complex due to subjacent prisoner's dilemma. The problem is indeed to minimize the downloading time for each user and to maximize the probability that the file stays in the network even for low request frequencies. The former is an individual optimization but the latter benefits to the whole community, and both optimizations are made with detriment of the other. To simplify the problem we will make the following assumptions:

- Most models are download-oriented: the peers try to download blocks they need from peers or seeds they know. We choose an upload approach, where peers and seeds decide which block they upload and to whom it is uploaded. Even if it does not exactly reflect the reality, we think it makes strategy clearer without altering practical activity.
- Everyone that can upload a block to somebody will do so. Once again, reality is more complex ([1] uses a choking algorithm to stimulate the exchanges), but we have a good approximation. Peers and seeds often choose a maximum upload bandwidth and a maximum number of connections and stay stuck to these maxima. As we show in section 4, this leads to an interesting result, called "torpor" where upload can sometimes severely injure a torrent.

3 First simulations

In this section, we suppose S and P are fixed. It can be interpreted as the worst case of peer behavior: a given number of philanthropic seeds are opposed to greedy peers that leave as soon as their download is finished and are instantly replaced by other greedy peers waiting in a queue. We want to study how that sort of torrent depends on the seeds. In other word we want to investigate the chance that the networks keeps distributed copies of the file even if no seed is present.

3.1 Strategies

The state of a torrent at a given moment can be represented by a logical $T = (t_{p,k})_{\substack{1 \leq p \leq P \\ 1 \leq k \leq K}}$ matrix where:

$$t_{p,k} = \begin{cases} 1 \text{ if user } p \text{ possesses the block } k \\ 0 \text{ otherwise} \end{cases}$$

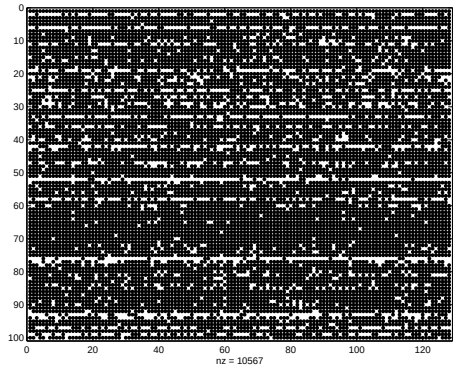

FIG. 1 – *Stable state in a globally random upload strategy*

Anytime a block upload is finished, according to our assumptions, the uploader must choose another couple (peer,block) and begin another upload. Strategies in our model resume in strategies in the choice method. We propose the following strategies:

Globally random strategy In GRS, each uploader chooses a couple (p,k) at random.

Block then peer decomposition The uploader chooses the block it will upload, then the peer that will receive it.

Peer then block decomposition Inverse as above.

Decompositions strategies vary with the sub-choice method. We propose two basic methods: uniformly random choice, that is self-explicit, and positive discrimination choice, where you choose the rarest block or the poorest peer in term of download progress [2]. For simplicity, we call decomposition strategies by initials. For example, BRPD strategy means a *B*lock is *R*andomly chosen, then the *P*eer is selected using positive *D*iscrimination.

3.2 Results

We choose $S = 1$ and $P = 100$ and $K = 120$. Everyone has the same upload bandwidth, so we can say the time to upload one block is the time unit. The ratio download/upload r is infinite [3]: the number of blocks a peer can get in a time unit is not bounded.

Simulations using the GRS starting with P empty peers (the deployment phase) tend towards two different stationary states, that we will call *safe* state and *torpor* state. Both states are roughly equiprobable. In safe state, seeds are unnecessary and the peers suffice themselves for themselves to keep the torrent alive. In torpor, there is a block possessed

2. We remark that a decomposition strategy that uses random in both choices is not equivalent to the globally random strategy.

3. Actually, for most people using ADSL, this ratio is 4 or 8. Later, we will give results for such ratios.

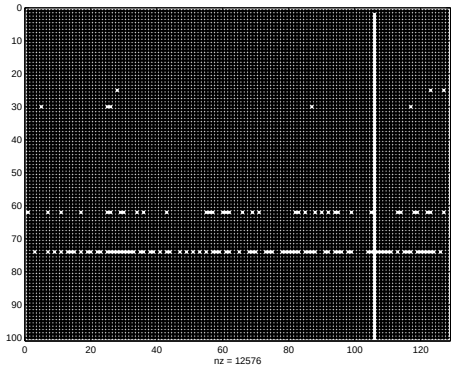

FIG. 2 – *"torpor" state in a globally random upload strategy*

only by the seed (most of the time no peer possesses it), all the peers are waiting forever for the missing block and contaminating the newcomers. If the seed quits in a torpor state, the torrent dies. Figure 1 shows the matrix T in a typical stable state and figure 2 shows a typical torpor state (lines represent peers; line 0 stands for the seed).

Other strategies converge towards either a stable state or a torpor state:

– BRPR and PRBR converge towards a torpor state.

– other strategies tends towards a safe state.

We note that all the safe states are not identical. Parameters like the average block density or the download speed can vary, as discussed in section 5.

4 "torpor" characteristics

Torpor is a dangerous state where the torrent can not live without seed. This is why we want to deepen the whereabouts of this state.

4.1 *Torpor* apparition

The appearance of torpor in the deployment of a torrent is intuitively easy to understand. All safe states imply all the blocks having on average the same density (as shown by figure 3). On the contrary, torpor means having a very rare block and the others almost totally spread. In the growing phase, the obligation of upload leads to a geometric progression of every block uploaded by the seed(s). This leads to strong irregularities of the block distribution in the earlier times. If this irregularities are not correctly smoothed by the subjacent upload strategy, the death of the torrent can occur during the earlier cycles. This is why BRPR and PRBR strategies always lead to torpor and positive discriminant strategies always lead to safe states: the first ones do nothing about the block distribution while the last ones are all

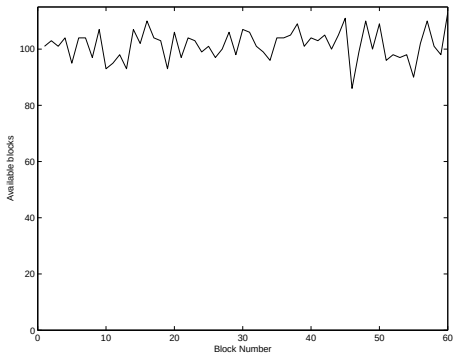

FIG. 3 – *Population of each block in a stable state*

about equity. We can then wonder why a safe state can be reached using GRS. Intuitively, the answer is that GRS is partially positive discriminant: the probability for a block k to be downloaded is basically proportional to the number of peers needing this file, and the probability for a peer p to receive a block is roughly proportional to the number of blocks it needs.

4.2 *Torpor* robustness

In *torpor* state, we call "patients" peers having all the blocks but the missing one. Because of the contagion of the patients, a torpor can be irremediable even using the smartest strategy, assuming peers that are able to upload do so. For example, if $S = 1$ and $r = +\infty$, a *torpor* will be stable as long as $P \geq K$. The time to "heal" a patient is then greater or equal than the time to contaminate a newcomer. This result spreads with r and S unspecified, and a sufficient (but not necessary) condition for stability, independent from the strategy, is:

$$P \geq S + S(K-1)(1+\frac{1}{r}) \tag{1}$$

Proof A torpor is necessary stable when the contaminating power of patients is greater than the "healing" power of seeds. The number of newcomers patients can contaminate in a time unit is $\frac{P'}{K-1}$, where P' is the number of patients. Seeds can heal at most S patients per time unit. Healed patients (newcomers) need $\frac{K-1}{r}+1$ time units to be recontaminated.

As long as $\frac{P'}{K-1} \geq S$, the torpor is stable (note that P' can vary at each time unit). The healing bound implies $P - P' \leq S(\frac{K-1}{r}+1)$. These two inequalities lead to (1).

■

With the assumption *everyone that can upload a block to somebody will do so*, patients will always perform good strategies to contaminate other peers. On the other hand, the healing

strategy of seeds must be highly precise to be efficient (each seed must heal a patient with each time unit). Thus we can say that (1) is a precise stability bound for strategies using positive discrimination first (whether it is on the peers or the blocks), while torpor stability in random oriented strategies is much more important.

4.3 The missing block in real networks

Although our model is upload-oriented, it captures a phenomena that occurs in real peer-to-peer sharing networks. For now, we can only give intuitive reasons for that. In eDonkey networks, users trying to get a file are waiting in a queue, and once their turn comes, they are granted one (or more) blocks. As far as we know, the choice of the block is random (except possibly for the first and the last block, for previewing purpose). The queue system is *a priori* independent from the number of blocks users possess (although it may be possible there is a slightly negative discrimination due to the fact that "old" peers are more likely to have at the same time many blocks and good positions in queues). Thus we would say eDonkey transferts can be seen as a RPRB strategy. Knowing how this strategy is sensible to torpor phenomenons, we may have an interpretation of torpor in eDonkey.

For Bittorrent networks, apparition of torpor seems more surprising, as the strategy is globally a block-oriented positive discrimination. Then again, we can only rely on empiric observations to suppose torpor can happen when P tends toward 0 (after a first rush, the torrent becomes less attractive). In future work, we will introduce a flexible processus for arriving of newcomers and try to verify this hypothesis. However, equation 1 shows that when a torpor occurs, healing can be fastidious or even impossible. A frequent strategy for the original seed of a torrent (the user that first offered the file to be shared) when downloads are stopping is to re-seed, that is to reintegrate the torrent the time for new seeds to appear, then to leave. The problem is real patients are often almost as miserly as the patients of our model. That means they stay in the torrent a few moments after the download is complete, then leave. Then we can imagine the following situation: after a torpor, the original seed decides to reintegrate the torrent. Rapidly, many patients become "seeds", so the original seed quits believing the torrent is alright. But if the torpor is not completely healed, the odds are high for the remaining patients to contaminate the torrent again once the "seeds" are gone.

Importance of the last block

We can wonder if it is that important if there is a block missing. With Error-Correcting Codes (ECC), it is possible to share files that can be completed without all the blocks. But ECC alone can not solve the problem for long: knowing a missing block is acceptable, peers will start to behave consequently. So ECC allow a torrent to function in torpor. We just consider a virtual torrent with $K - 1$ blocks. And what happen if this virtual torrent goes in torpor?

The conclusion of this section is that the missing block is a real issue in P2P networks today, and that a real optimization of the transferts strategies can be beneficial.

5 Efficiency of upload strategies

We now want to compare the different strategies seen in 3.1 on other domains than stability, such as average download speed, original diffusion, density, and robustness to *very greedy peers*.

5.1 Average download speed

The global download speed is bounded by the sum of upload bandwidths. For a (S,P) torrent with uniform upload bandwidth U, the average download speed can not be greater than $\bar{D}_{max} = \frac{S+P}{P}U$. Simulations show that whenever a safe state is reached, average download speed tends towards this limit.

If the torrent goes in torpor, average download speed can be lessened: it is a good approximation to say that only seeds can trigger the end of a download. Thus if there is one seed and if the theoretical minimum average time between 2 finished download is inferior to a time unit, peers are going to stay idle (without uploading) part of the time. More precisely, with S seeds, the average download speed is bounded by $\min(\frac{S.K}{P}\bar{D}_{max},\bar{D}_{max})$.

5.2 Speed of Deployment

The deployment is the very dangerous phase of a torrent. If the original seed leaves before it ends, the game is over. Moreover, the original seed often wants to minimize its upload time. For both reason, deployment must be fast. The minimum time for deployment is the time for the original seed to upload each block one time, that is K time units. It is achieved if a positive discrimination on blocks is used.

5.2.1 Linear strategy

Sharing a file linearly (from the first to the last block) like KaZaA protocol is not a good idea from a torpor point of view. Even if peers do not quit as soon as they get their last block, the last blocks of the file are very likely to miss sooner or later. The quality of deployment in a linear strategy is also far from optimal, whatever the peer strategy is:

- If the original seed allow every peers to download its block, deployment will take $P.K$ time unit (assuming all peers are treated equally).
- The original seed can restrain the peers allowed to download from it to a subset of size $Q < P$, so time for deployment will be $Q.K$ time units.
- In KaZaA networks where you cannot share a file until you completely possess it, achieving the minimum deployment times implies to upload to only one peer. If the original seed quits after that, you come back to the original state, except the original seed is know a new seed whose reliability is unknown.

5.2.2 Random block strategies

Strategies where blocks are chosen at random are not really better. In fact, the more K is important, the more the original seed has to upload, as shown by the following theorem:

Theorem 1 *Given a set $F = \{1,...,k\}$, the mean time to choose one block of each in a random choice repeated in F is equivalent to $k\ln(k)$.*

Proof let T_k be the stopping time (in number of draws) when in the sequence of number chosen each symbol 1,2,...k is at least one time. Then

$$\mathbb{E}(T_k) = \sum_{\omega} T_k(\omega)$$

Let's look the evolution of the process:

1. a first number is chosen (with probability 1)
2. with probability $(1/k)^{i_1}\frac{k-1}{k}$ it is chosen i_1 times before other one is taken.
3. with probability $(2/k)^{i_2}\frac{k-2}{k}$ one of both is chosen i_2 times before an other one is taken.
4. ...
5. with probability $(1-1/k)^{i_{k-1}}\frac{1}{k}$ one of the $k-1$ last ones is chosen i_{k-1} times before the last one is taken.

for one of those trajectories $T_k = k + i_1 + i_2 + ... + i_{k-1}$. Thus

$$\begin{aligned}
\mathbb{E}(T_k) = \sum_{i_1...i_{k-1}} \Big(& (k + i_1 + i_2 + ... + i_{k-1}). \\
& (1/k)^{i_1}\frac{k-1}{k}...(1-1/k)^{i_{k-1}}\frac{1}{k}\Big) \\
& = \frac{k-1}{k}...\frac{1}{k}\sum_{i_1...i_{k-2}} (1/k)^{i_1}...(1-2/k)^{i_{k-2}}. \\
& \Big(\sum_{i_{k-1}} (k + i_1 + i_2 + ... + i_{k-1})(1-1/k)^{i_{k-1}}\Big) \\
& = \frac{(k-1)!}{k^{k-1}}\sum_{i_1...i_{k-2}} (1/k)^{i_1}...(1-2/k)^{i_{k-2}}. \\
& \Big((k-1+i_1+i_2+...+i_{k-2})\frac{1}{1-(1-1/k)} \\
& + \sum_{i_{k-1}} (1+i_{k-1})(1-1/k)^{i_{k-1}}\Big)
\end{aligned}$$

Notice that

$$\sum_{i\in\mathbb{N}} (i+1)X^i = \frac{d}{dX}(\frac{1}{1-X})(X) = (\frac{1}{1-x})^2$$

it also true in $\mathbb{R}$ for $X = x$ with $|x| < 1$. Thus

$$\begin{aligned}\mathbb{E}(T_k) &= 1 + \frac{1}{1-1/k} + ... \frac{1}{1-(1-1/k)} \\ &= k(1/k + 1/(k-1) + ... + 1/1) \\ &= k(\ln(k) + \gamma + \epsilon(k))\end{aligned}$$

with $\epsilon(k)$ tending towards 0 and γ is the Euler constant.

■

5.3 Density

In Bittorrent, the number of distributed copies seen and the average download are often considered as indicators of wealth. As said before, each block in a safe state has roughly the same density, so the two parameters are basically proportional (see figure 3). From this point of view, peer-oriented discriminant strategies seem to be better, as most of the earlier blocks are possessed very fast. But as we see in next paragraph, having the biggest density is not always a good thing.

5.4 Robustness to *very greedy peers*

Very greedy peers (VGP) are peers wanting to get their file as soon as possible and that are likely to trick the torrent to do so.

5.4.1 VGP and positive discrimination

Positive peer discrimination helps new users to get blocks faster. It increases the probability that the uploaded blocks can be uploaded many times. By construction this leads to stable states where the repartition of blocks has a small standard deviation (in PDB(DR) it is close to 0). Intensity of users arrivals is maximal, because users can upload almost every time. The problem is that the peers wait a very long time at the end to get their last block (see figure 4 for download speed during download progress), and that it is profitable to cheat by announcing that you have to download a number of blocks larger than your real one. With $K = 60$ and $P = 120$ (parameters of figure 4), declaring 2 missing files instead of 1 increases the average download speed by 86, thus increasing the average effective download speed of the last block by 43.

5.4.2 VGP robust strategies

Let $f(k)$ be the average speed download of peers possessing k blocks. A strategy is robust regarding VGP if asking for blocks you already have is not benefic.

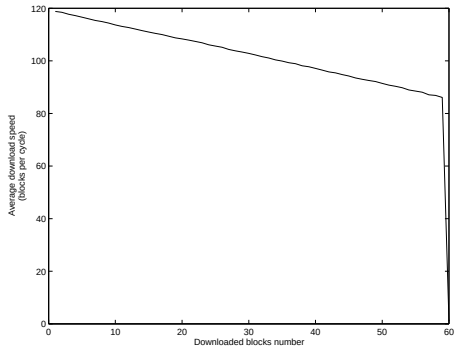

FIG. 4 – *Average download speed*
the download progress in a positive peer discrimination strategy

A random-block strategy is VGP-robust if f has the following property:

$$\frac{f(k)}{K-k} \text{ increases} \tag{2}$$

Proof A peer possessing k_1 blocks and declaring $k_2 < k_1$ blocks will have a average download speed of $f(k_2)$ with a proportion $\frac{K-k_1}{K-k_2}$ of useful blocks if the block strategy is random. Thus lying is not profitable if $f(k_1) > \frac{K-k_1}{K-k_2} f(k_2)$. This is the case if the function $k \to \frac{f(k)}{K-k}$ increases.

■

We remark that this result stay true in a discriminant-block strategy if the state is safe, because all the block have roughly the same density. In a torpor state with seeds, this result is false: lying allow the VGP to get the missing block faster than expected, with heavy cost for the torrent (one of the few who possesses the missing block leaves sooner).

We verify easily that positive peer-discrimination does not fulfill (2) for $k = K - 1$. Contrary, in GRS, the download speed (in a safe state) is basically an affine function of k (see figure 5) that fulfill (2). This linearity can be intuitively explained if we arbitrary change the number of blocks k a given peer p_0 possesses (all other parameters being unchanged). The average speed download for p_0 will be obviously proportional to the $K - k$ missing blocks. Unlinearities noted in figure 5 comes from random fluctuations and retroactions that have been neglected.

The conclusion of the study of VGP is that positive peer-oriented strategies are not robust, and that it is better to sacrifice some density and homogeneity (see figure 6 for a typical download distribution in GRS) to gain robustness. Especially when that does not harm the global download speed, as we have seen.

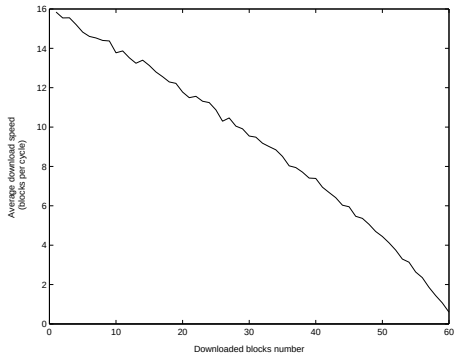

FIG. 5 – *Average download speed in the download progress : GRS-stable case*

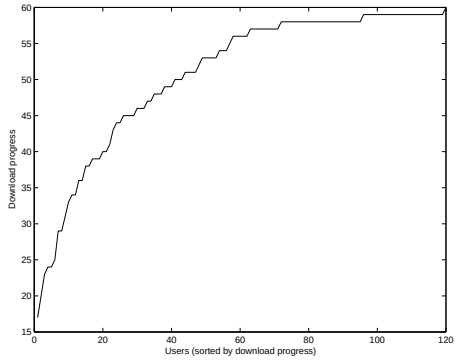

FIG. 6 – *Repartition of download progresses among users in GRS*

6 Future work

We think it is possible to study the stability of GRS and the probability of torpor during deployment using a mean field theory. That should give a base to purpose new strategies. We also want to improve our model using variable upload bandwidth and departing and arriving of peers ([7] gives precious information about real distributions). This will allow to make our model more accurate and to verify some hypothesis such as the apparition of torpor during the decline of the torrent and the difficulty to reseed. Lastly, we think about analyzing logs from eDonkey servers and Bittorrent trackers to validate our model.

7 Conclusion

We gave a rather precise and intuitive survey of missing block issues. We showed that a block-oriented discriminant strategy is more efficient than a random strategy, so we could say Bittorrent behaves better that eDonkey from this point of view. Lastly we saw that peer-oriented discrimination is less important, it can even be damageable regarding to bad social behavior. These results could be used to enhance existing protocols. For example, a tracker aware of torpor issue could anticipate it and reseeding could be more effective.

Références

[1] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[2] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks, 2002. in The ACM SIGCOMM'02 Conference, August 2002.

[3] Fabrice Le Fessant and Simon Patarin. Mldonkey, a multi-network peer-to-peer file-sharing program, April 2003.

[4] N. Harvey, M. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties, 2003.

[5] J. Kangasharju, J. Roberts, and K. Ross. Object replication strategies in content distribution networks, 2001.

[6] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM 2001*, 2001.

[7] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, San Jose, CA, USA, January 2002.

[8] Ion Stoica, Robert Morris, David Karger, M. Francs Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.

[9] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399